## ABSTRACT OF THE DISCLOSURE

By encoding an exception triggering value in storage referenced by an instruction in an otherwise unused slot (e.g., the delay slot of a delayed control transfer instruction or an unused instruction position in a VLIW-based architecture) coinciding with a safe point, an efficient coordination mechanism can be provided for multi-threaded code. Because the mechanism(s) impose negligible overhead when not employed and can be engaged in response to an event (e.g., a start garbage collection event), safe points can be defined at call, return and/or backward branch points throughout mutator code to reduce the latency between the event and suspension of all threads. Though particularly advantageous for thread suspension to perform garbage collection at safe points, the techniques described herein are more generally applicable to program suspension at coordination points coinciding with calls, returns, branches or calls, returns and branches therein.